Express Mail Label No. EL474251554US

Attorney Docket No. 10334-1US

TITLE OF THE INVENTION

AUTOMATED SYSTEM FOR ANALYZING CHARGES AND CREDITS OF BANKS AND OTHER FINANCIAL INSTITUTIONS

COPYRIGHT NOTICE AND AUTHORIZATION

BACKGROUND OF THE INVENTION

Over the last few decades, banks and other financial institutions, hereafter collectively referred to as "banks," have begun charging their clients for the services performed. These charges include, but are not limited to: deposit ticket charges, night bag deposits, electronic transmittal of information, returned check charges and postage. Banks no longer rely to the same extent as they have in the past on the income generated from loans. Bank officers have even attended seminars strictly targeted to increasing revenue through service charges. Even if the evolution of service charge revenue was unintentional, it has become a lucrative opportunity for many banks.

Clients often receive cryptic monthly statements as to what charges they are actually paying for. Consider a scenario wherein supermarket aisles did not indicate the price of the food on the shelves. In addition, suppose the receipt did not include a detail of the food that was purchased or the price of each item. Instead, only a total due during that shopping trip was provided. Or, if the detail was included, it was not listed in readily understood terms, but only in the UPC Barcodes of the product. It would not be easy to comparison shop, or to determine better ways to shop, such as buying larger sizes of items you may use frequently. This scenario is analogous to the way service charges are presently handled between banks and their clients. Furthermore, the bank often deducts

service charges from the earnings credit (Interest) posted to the clients' accounts. To continue the supermarket analogy, imagine if the receipt didn't tell you how much you actually spent that day, just a running total against some prepaid balance you may have on account with the supermarket. Banks may not have intentionally created these problems. However, these problems prevent clients from understanding and managing their charges effectively.

These analogies illustrate how many large businesses (clients) currently interact with their banks. Coupled with the fact that charges imposed are not listed in layman's terms on the bank analysis, it has become extremely difficult for a client to control these charges. This weakens the client's financial and bargaining position and gives little leverage when negotiating with the client's bank. It also inhibits the client from attaining the highest possible earnings credit.

Many clients have numerous locations (stores) throughout the country who use the same bank. The statements issued by the bank offer little help in determining charges per store, and do not provide the client's financial managers with the essential data required to make appropriate fiscal decisions.

An additional problem for clients is that each bank uses its own set of terms to define its service charges and credits, thereby making it difficult or impossible for a client to comparison shop among banks for the best deal on service charges and credits.

Accordingly, there is an unmet need for analysis tools to allow a client to more fully understand bank service charges and credits. The present invention fulfills such a need.

SUMMARY OF THE INVENTION

A first embodiment of the present invention provides a computer-implemented scheme for preparing bank service charge reports for banking activity of a client. In the scheme, an electronic translator is provided that converts service charge items of a plurality of individual banks, expressed in terminology of the respective individual banks, to a service description expressed in a standardized terminology. At least some of the service charge items of different banks have the same standardized service description. Bank service charge items and bank account data are inputted into a computer. The bank service charge items and bank account data are for a specified period of time for one or

more banks being used by a client. The bank service charge items and bank account data are obtained from one or more bank statements of a client and are expressed in the one or more statements in terminology used by the one or more banks. In the computer, one or more bank service charge reports are automatically created for the client from the inputted bank service charge items by using the electronic translator. Each bank service charge report provides a breakdown of bank service charges based upon the standardized service descriptions.

A subset of the standardized service descriptions of bank service charges may define a total amount of a predefined type of bank service charge costs. In this scheme, the total amount of a predefined type of bank service charge costs is calculated by adding together the bank service charges in the subset. A total cost report is then created for the predefined type of bank service charge costs from the calculated total. The subset may define total depository costs, total check costs, total account maintenance costs, or any total of a service charge category.

A historical database may be provided of average bank service charges for selected standardized service descriptions based upon bank service charges of a plurality of clients. The individually broken down bank service charges may then be compared to the average bank service charges. The comparison may be used by the client to identify potentially excessive bank service charges. Alternatively, a database of bank service charges may be provided for each of the standardized service descriptions based upon bank service charges of all of the banks used by the client. The individually broken down bank service charges for one of the client's banks may then be compared to the bank service charges for other banks used by the client. Again, the comparison may be used by the client to identify potentially excessive bank service charges.

One of the standardized service charge items may be deposit ticket costs, wherein deposits have a predefined earnings credit rate. The deposit ticket costs and the earnings credit rate may be used to calculate a breakeven deposit amount wherein the earnings credit exceeds the cost of depositing money.

In a second embodiment of the present invention, a similar process as described above is performed for earnings credit items.

BRIEF DESCRIPTION OF THE DRAWINGS

The file of this patent contains at least one drawing executed in color. Copies of this patent with color drawings will be provided by the Patent and Trademark Office upon request and payment of the necessary fee.

Fig. 1 (provided in parts 1 and 2) summarizes reports generated by the present invention, and the information provided by the reports that is not currently available to a banking customer;

Figs. 2-14 show examples of the reports summarized in Fig. 1;

Fig. 15 is a schematic block diagram of a cross-reference table in accordance with the present invention;

Fig. 16 shows an entry screen of a form used to collect and store service charges;

Figs. 17-19 show the contents of tables used in the cross-reference table;

Fig. 20A and Fig. 20B show screen displays for viewing bank description and service description data in the cross-reference table;

Fig. 21 shows a table that stores client data;

Fig. 22 is a flowchart of a history compilation process of the present invention;

Fig. 23 shows the contents of a Bank Service Charge History table used for the history compilation process;

Figs. 24 and 25 shows the contents of tables used in a dynamic compilation process;

Figs. 26 and 27 are data entry screens used in the report generation process;

Fig. 28 shows the contents of a Category table which is used to generate the report in Fig. 7;

Fig. 29 is a screen display for selecting categories that the user wishes to appear in a report;

Fig. 30 is a screen display for selectively choosing any combination of clients, banks, and/or service charges that the user wishes to appear in a report;

Figs. 31 and 32 show excerpts from charts that are used to determine costs of a particular service charge category; and

Fig. 33 is a screen display of a Working Sheet Form that is used to summarize service description items, and their actual costs and potential cost savings.

DETAILED DESCRIPTION OF THE INVENTION

Certain terminology is used herein for convenience only and is not to be taken as a limitation on the present invention. In the drawings, the same reference letters are employed for designating the same elements throughout the several figures.

DEFINITIONS

Comp Balance - balances on hand at bank, subject to the bank's earnings credit.

Earnings Credit – interest earned on available client's assets, taken after the Federal Reserve requirement is deducted.

Federal Reserve Requirement – amount by law which must be reserved by the bank for the Federal Reserve (non-earning asset), usually 10% of client's balance.

Treasury Bill – (T-Bill) – interest rate set by the US Government.

Fed Funds Rate - A target interest rate for banks borrowing reserves among themselves. It is set by the Federal Open Markets Committee. See web site: http://www.bankrate.com/brm/green/define/#fomc for more information.

Float – checks which have been deposited but funds are not yet available.

Revision Date – date of the bank statement (usually issued on monthly basis).

Basis Point - one one-hundreth of a percent, e.g., 300 basis points is 3%.

OVERVIEW OF PRESENT INVENTION

The present invention is implemented as a software program called "The Bank Analyzer," that aims to empower clients with the necessary information to ensure fair treatment from their bank. The software generates numerous reports which explain in layman's terms exactly how bank service charges are broken down. Furthermore, the reports are presented in color, making it easy for even a novice to elicit the needed

information. Armed with such information, a client will understand exactly where and how service charges can be reduced and/or earnings credits can be increased.

Prior to the present invention, some clients may have suspected that certain bank charges were askew, certain transactions were wasteful, and certain transactions were unnecessary. However, there was never any solid clear-cut method of substantiating these suspicions. Furthermore, it was the belief of clients that the resources necessary to even begin an investigation would outweigh any results. The present invention includes a plurality of analysis techniques that were previously not available and which are implemented in formulas that generate Illumination Reports (described below). The Illumination Reports yielded by the present invention provide a catharsis to the clients in managing their bank charges.

The basic philosophy of the invention is (1) Effective Management of Bank Service Charges; (2) Illumination of Data; and (3) Edification of the Client.

In the examples provided in the figures and tables below, Bank Names and Client names are fictitious, but the values and dollar amounts represent actual data.

Illumination Reports Table (Diagnostic Section)

Fig. 1 (provided in parts 1 and 2) summarizes in table form each report from the diagnostic section of the present invention, highlighting the areas of most impact. In many cases, clients are unaware that the data produced by the present invention may be significant to the financial well being of their company. In other cases, there is no simple method of obtaining this data. Lack of data precludes effective management of such charges.

The disclosure below covers various methods and systems to assist clients in managing their bank service charges and reducing or eliminating cost incurring transactions. The focus also centers on elevating client awareness and providing a clear-cut analysis of how bank charges are affecting their bottom line. Also, the disclosure presents the algorithms of the software which provide this assistance. Exemplary methods include:

1. Revealing actual Bank Service Charges assessed.

2. Minimizing strap charges, i.e., the charges a bank assesses for counting clients money, by strapping in denominations which yield the best results.

3. Comparison of clients' bank charges, i.e., Deposit Tickets, Night Bag Deposits, etc... with charges of other clients' banks nationwide. Recommendations are made based on these national averages.

4. Evaluation of clients' non-earning assets and recommendations on how to maximize revenue from these assets.

Exemplary algorithms include:

1. Using dynamic report generation forms to allow the user to extract the necessary information to execute the methods.

2. Creating cross-reference tables between a bank charge and its description according to the particular bank, and the actual charge expressed in layman's terms.

3. Storing all bank analysis data in a minimum number of tables, and calculating and deriving comparison information "on the fly," that is, each time a report is generated. This ensures up-to-the-minute accurate information.

DETAILED DISCLOSURE

This section outlines what each report accomplishes. Refer to the attached figures for examples that correspond to the figure numbers listed. The alphanumeric characters refer to data highlighted in the respective figure. For example, "2a" highlights data in Fig. 2.

Bank Balance Data (Fig. 2) – Here the Bank Analyzer exposes the total service charge assessed a client for the particular revision date. (2a) Note that the Bank Analyzer reveals the actual charge, not the 'net' one which the client now sees. The net charge includes a 'comp' balance, which are client's balances subject to the Banks' earnings credit. In some cases, the earnings credit offsets any service charges. That is, the client believes that no service charges are being paid, when, in fact, the service charge is deducted from any earnings credit due. Fig. 2 shows how this client (SUPER-CON Convenient Stores) has paid $57,192.50 in the month of October, 1999 to their banks. The report details not only totals but amounts to each bank, e.g., $15,035.17 paid to the Western Savings Co. (see line #2 of the example report)

Earnings Credit Analysis (Fig. 3)

This report color codes basis point shaved, to indicate abnormal bank activity.

This report is a scorecard to see how a client's bank performs against the T-bill rate. (3a) It indicates the difference between T-Bill and client's actual earnings. The results are instantly revealed. This process would consume hundreds of man-hours to produce otherwise. Armed with this report, a client can be sure to be credited with at least the T-Bill Rate.

Fed Funds (3b) – This same report also shows the amount of money the bank is earning by investing the client's money at the Fed Funds rate. The bank will often invest in Fed Funds when they are not using it to generate loan revenue. This gives the client a look at how much money the bank can earn from the client's money.

Income Difference (3c) – Finally, the income difference is determined, which is the difference a client can add to its bottom line if given the corresponding interest rates. This generally results in a tremendous increase to the client's bottom line. (3d)

The Bank Names, as well as the basis points shaved, are printed in degrees of color. This facilitates isolating the cases where the client can realize the maximum benefit. (3e)

Earnings Credit And Negative Collected Rates (Fig. 4)

As a supplement to the Earnings credit Report, this report matches the Earnings Credit on positive balances to the rate imposed on Negative Collected funds. Clients should be charged the prime interest rate. However, typical findings indicate that clients are sometimes charged as much as 300 basis points over prime. A fair bank would actually pay at a 1:1 ratio by posting earnings at the same rate imposed for the negative collected balance. The example report shows one sample bank (4a) which is charging significantly higher negative collected rates than earnings credit rates, and one sample bank (4b) which is charging fair rates.

Checks, Deposits And Float Data (Fig. 5)

Avg Check Size Per Store (5a)

Occasionally, the bank will assess a float table that does not correspond to actual availability. This report will alert client to this fact, by showing them the Avg Check Size per store. The client usually has a good idea of the average check a store receives, (e.g., a drug store may receive checks that average about $75, whereas a lumberyard may have

checks averaging $300). If this check size becomes inflated, that is, if the client sees that the value is more than it normally would be, then the client knows to investigate and to look at each check. In many cases, the client may realize that the bank is not posting the checks to his/her account promptly. Without the use of the present invention, discovery of this fact would consume inordinate amounts of time and resources.

Out Of District (5b)

The Out Of District column indicates checks received from customers of the client that are drawn on banks outside the local district of the depository bank. It is common that most customers bank locally. Therefore, 90% should be in district. The column indicating 2 Day % is the derived Out Of District number. If this number is not within this normal average, it is an alert that the bank may not be providing the proper calculation with the float. Also, it is likely that the service charge is increased, because Out Of District service charges are generally more expensive.

Checks On Us (5c)

This column determines the presence that a bank has in a region, and On Us Checks should be treated as cash and should be made available immediately. Clients can now ensure that this is the case.

Breakdown By Service Charge Groups (Fig. 6)

The Bank Balance Data report illuminated the actual Total Service Charges assessed. This report begins putting the Total Service Charge number under the microscope, commonly referred to as drill-down reporting in the computer field. This report shows the 3 main categories:

Categories Depository Costs (dep tickets, cash bags, etc...) (6a)

Check Costs (Checks dep, ret items, etc..) (6b)

Account Maintenance (balance reporting) (6c)

The percentage of the total cost is analyzed, and if these three categories do not total close to 100%, then there is a good chance that a particular charge is askew. Potential items to investigate include:

1. Negative interest charge for using uncollected funds (See Earnings Credit and Negative Collected Rates report)

2. Bounced check fees - too high?

These percentages can be compared with other clients' percentages to assist in evaluation. Volumes (amount of activity) are also considered, to ensure fair comparisons.

Breakdown By User Selected Groups (Fig. 7)

This report allows for the creation of custom reports tailored to each client. Multiple service charges can be lumped into groups, and then these groups can be evaluated. For example, a client may want to see a breakdown of lock box deposit charges. Presently, this would require mining every statement for any charge pertaining to a lock box. It is not always clearly indicated on the bank statement, so research could also be required. Using a conventional analysis process, it could require numerous hours or more for a client to presently determine all of the lock box charges. In the present invention, service charges can be grouped into a category, for example, Lock Box Charges. Then, the cross-reference table links all lock box charges: Correspondence, Courier, Delivery US Mail, Checks Deposited, etc... The report then displays at a glance the actual Lock Box charges that a client is paying for.

Lock Box Charges is just an example category. Any group can be analyzed, e.g., Depository Costs, Checks, Cash, Coin, etc... This information is invaluable to the client and has never before been available to the client. Any categories can be supplied, yielding hundreds of possibilities for analysis depending on the client's particular situation.

Unit Price Summary (Fig. 8)

This report shows every service charge assessed for the client encompassing all stores, all banks, and every line item. The report shows at a glance monthly totals, yearly totals (calculated) and also average and weighted average, which considers volume. This information was so difficult to obtain using conventional analysis processes that clients never even considered attempting to acquire it.

Deposit Ticket Costs (Fig. 9, provided in parts 1 and 2)

This report examines the prudence of daily deposits by examining every cost associated with making a deposit. Since clients make frequent deposits of large sums of checks and cash, these charges can be substantial. See 9a for a sample of the total cost incurred in one year for one client. Prior to the present invention, these charges could

never be investigated or verified, and therefore it was not possible to show inefficiencies in the client's business practices. This report also determines the daily deposit a client must make in order to "break even." It is actually possible for a client to lose money by making too many deposits, because the cost of depositing the money outweighs any earnings credit that may be earned. (9b)

Finally, the Potential Daily Loss columns show the exact amount lost (9d) if the deposit were equal to the column headings. (9c) It should be noted that these values depend upon the average return rate a client expects to make. (9e) This value is dynamic, and can be changed (inputted) each time the report is generated.

Cash Activity Costs (Fig. 10)

Banks charge their clients to count the cash being deposited. Clients are generally charged in one of two ways: by strap or per $1000 counted. If a particular client receives a lot of cash, it behooves a client to be charged by strap since 1000 single dollar bills is charged the same as 10 one hundred dollar bills, even though it is more labor intensive to count 1000 bills than 10 bills. When charged by strap, 100 100 dollar bills are charged the same as 100 1 dollar bills. This decision varies depending on the cash composition of the business client is engaged. For example, a newsstand which receives numerous one dollar bills is generally better off paying by the 1000, because they will have numerous bills. On the other hand, a supermarket may be better off paying by strap, as they may receive greater denominations. In either case, the reports will highlight this information, permitting the client to choose wisely.

Another area often overlooked by clients is the Rolled coin charges (10a). Clients are often charged for the phone call to order the coin! This report allows deeper probing into this matter.

Another benefit of the Cash Activity Costs report is that it allows for verification that the proper amount of cash is being counted. Currently, clients must assume that the banks' report is correct.

To gain insight into just how much it costs to count cash, see the client's monthly charge (10b) and yearly projection. (10c)

Banking Activity (Fig. 11)

This is a summary report, which details vital information such as Deposit Tickets, BAI Detail, Night Bags and Rolled Coin charges. It is used as a support and guide to aid the client in understanding the detail diagnostic reports.

Analysis By Unit Price (Fig. 12)

This report takes the Unit Price Summary and drills down every charge to list each bank where the charge was derived. It is a lengthy report and is used to isolate problem areas when the summary indicates a potential problem area. It serves as a complete audit trail tracking the lowest level of detail from each of the client's banks.

Illumination Reports (Action Items) – (Figs. 12-14)

This section provides a 'blueprint' to the client for the steps which should be taken.

This section also contains every line item of every analysis from every bank for the client, and the revision date being considered. (see Fig. 12) This serves as supporting information when the client is making the decisions necessary to eliminate or reduce transactions, as well as negotiate with the bank for a reduction in charges which are deemed unfair.

This section contains the following items:

Working Sheet Report (Fig. 13) – contains a trace number which references where the original data came from. Provides client with actual volume and cost amounts, in addition to the recommendations based on the comparison with similar clients at similar banks. It also highlights the potential savings a client can garner from implementing these recommendations. Annual projections are extrapolated.

Projected Savings Report (Fig. 14) shows the clients' potential savings for each of their banks if the recommendations in Fig. 13 are implemented.

How the Bank Analyzer Works

Examples of source code for implementing the Bank analyzer are provided in this section.

There are 4 sections to the Bank Analyzer, as follows:

SECTION I - Static Compilation

SECTION II - Dynamic Compilation - Input of Monthly Bank Analysis Data

SECTION III - Report Generation - Black Box effect

SECTION IV - Illumination - Meeting with Client

This portion of the disclosure provides a detailed description of how the present invention works, and how it generates the reports to edify the clients.

SECTION I – Static Compilation

Static Compilation refers to the acquiring of data and inputting of data which will form the basis of the eventual evaluation prepared for the client. This section has two functions. The first function is to build a master cross-reference table, shown schematically in Fig. 15. The second function is history compilation.

Function #1 - Building A Master Cross-Reference Table

This function begins with the massive input of every possible service charge from every bank a client is associated. Each service charge is assigned a unique code and a layman's (layperson's) term. Fig. 16 shows an entry screen of a form used to collect and store service charges. Every possible charge incurred by the clients is entered here, and is assigned a standardized service description.

Fig. 17 shows a layout and structure of the cross-reference table. This table houses every bank service charge encountered from the statements. The ServiceCodes are a unique way of identifying all charges for all clients. This table can add new entries as banks create new service charges, but the information is generally static and therefore this table is generated during the Static Compilation section of the Bank Analyzer software.

Once the service codes are established, it is necessary to maintain a database of bank information and a link between the bank's service charges and the internal codes of the Bank Analyzer. This ensures that clients will see the same terminology when analyzing all of their banks, and will also provide the method of translating back to the bank's language when dealing with a specific bank. These functions are accomplished mainly by the use of two tables, Banks (Fig. 18) and BankServ (Fig. 19). Figs. 17 and 18 combine to make the cross-reference table. The Banks table houses in a computer

database each bank that a client is associated. The BankServ table houses in a computer database each service charge assessed by a bank, and the corresponding code assigned to this service charge by the Bank Analyzer software. The BankDescription field maintains the terminology used by the bank for future reference when dealing with the bank. This foundation data is collected in a computerized form and can be searched and retrieved via screen displays as shown in Fig. 20A and Fig. 20B.

Fig. 20A and Fig. 20B illustrate the use of an electronic translator that converting service charge items of a plurality of individual banks, expressed in terminology of the respective individual banks, to a service description expressed in a standardized terminology. The translation goes in the reverse direction when the results are used in discussions with the clients' banks. At least some of the service charge items of different banks have the same standardized service description. For example, the service charge for "Deposit Tickets" (standardized service code 00425) is referred to as "Deposited Processed" by Eastern Savings Co. (see Fig. 20A), and is referred to as "Credit Posted" by Commercial Bank (see Fig. 20B). In another example, "BAI Detail" (Bank Administration Institute Detail) (standardized service code 00973) is referred to as "Dep Recon Proc" by Eastern Savings Co. (see Fig. 20A), and is referred to as "Recon" by Commercial Bank (see Fig. 20B).

Fig. 21 shows a Clients table that stores client data for the Bank Analyzer software.

Sample Visual Basic for Applications (VBA) code for controlling interface of Banks table:

```
Code
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Form_Banks"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub Command16_Click()
D  Cmd.OpenRep  rt "Banks By Code", acViewPreview
End Sub

Private Sub C  mmand17_Click()
```

```
18  D  Cmd.OpenRep  rt "Banks By Name", acViewPreview
19  End Sub
20
21  Private Sub Command52_Click()
22  D  Cmd.OpenReport "Banks Quick Print", acPreview, , "[BankC  de] = '" & Me!BankC  de &
23  End Sub
24
25  Private Sub Form_Activate()
26  DoCmd.Restore
27  End Sub
28
29  Private Sub Form_Current()
30  If Me.NewRecord Then
31     BankCode.SetFocus
32  End If
33  End Sub
34
35  Private Sub Form_Load()
36  If Not IsNull(Me.OpenArgs) Then
37     Me!BankCode = Me.OpenArgs
38  End If
39  End Sub
```

Sample VBA code for controlling Service Codes interface:

```
Code
 1  VERSION 1.0 CLASS
 2  BEGIN
 3    MultiUse = -1  'True
 4  END
 5  Attribute VB_Name = "Form_ServiceCodes"
 6  Attribute VB_GlobalNameSpace = False
 7  Attribute VB_Creatable = True
 8  Attribute VB_PredeclaredId = True
 9  Attribute VB_Exposed = False
10  Option Compare Database
11  Option Explicit
12
13  Private Sub Command16_Click()
14  DoCmd.OpenReport "ServiceCodes By Code", acViewPreview
15  End Sub
16
17  Private Sub Command17_Click()
18  DoCmd.OpenReport "ServiceCodes By Description", acViewPreview
19  End Sub
20
21  Private Sub Form_Current()
22  If Me.NewRecord Then
23     ServiceCode.SetFocus
24  End If
25  End Sub
26
27  Private Sub Form_Load()
28  If Not IsNull(Me.OpenArgs) Then
29     Me![ServiceCode] = Me.OpenArgs
30  End If
31  End Sub
```

Function #2 - History Compilation

History compilation entails the process of taking bank statements from previous months and entering them into a computer database in the Bank Analyzer. Data collected

by this process is used for making accurate and informative comparisons between clients, and is helpful in determining appropriate and fair service charges. A flowchart of the process is shown in Fig. 22, and field layouts for a Bank Service Charge History table is shown in Fig. 23.

SECTION II - Dynamic Compilation - Input of Monthly Bank Statement Data

The Bank Statement issued by the bank to the client encompasses all stores for that client. The Bank Analyzer program provides a data collection form where this information is housed. This information is the raw material used to generate the final product. The two main tables used in this section are the AnalysisHdr and AnalysisDtl files. See Fig. 24 and Fig. 25 for the field layouts of these tables.

Sample VBA Code for controlling interface for AnalysisHdr:

```
Code
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Form_AnalysisHdr"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub BankCode_BeforeUpdate(Cancel As Integer)
If Not Me.NewRecord Then
   If MsgBox("You are about to change the bank code." & vbCr & "This will change all
service codes to be under the new bank." & vbCr & "Is this what you want to do?",
vbOKCancel, "Key Field Change") = vbCancel Then
      DoCmd.CancelEvent
      Me!BankCode.Undo
   End If
End If

End Sub

Private Sub BankCode_GotFocus()
BankCode.Dropdown
End Sub

Private Sub BankCode_NotInList(NewData As String, Response As Integer)
Dim Result
Dim msg As String

If NewData = "" Then Exit Sub   ' cleared combo box

msg = "" & NewData & " is not in the Banks file." & vbCr & vbCr & "Do you want t  add it?"
' Yes
If MsgB  x(msg, vbQuesti  n + vbYesNo) = vbYes Then
   DoCmd.OpenF  rm "Banks", , , , acFormAdd, acDial  g, NewData
End If
```

```
38
39  ' l   k for new record added
40  Result = DLo  kup("[BankCode]", "Banks", "[BankC  de]= '" & NewData & "'")
41  If IsNull(Result) Then
42     'suppress error message
43     Resp  nse = acDataErrContinue
44     MsgBox "Please enter a new customer number."
45  Else
46     ' they added it
47     Response = acDataErrAdded
48  End If
49  End Sub
50
51  Private Sub Command45_Click()
52  DoCmd.OpenForm "BankService Select", acNormal
53
54  End Sub
55
56  Private Sub Command46_Click()
57  DoCmd.OpenForm "AnalysisRpt Select"
58  End Sub
59
60  Private Sub Command47_Click()
61  Dim crt As String
62  crt = "[AnalysisNo] = " & Me!AnalysisNo
63  DoCmd.OpenForm "WorkingSheetHdr", acNormal, , crt, acFormEdit, , True
64  End Sub
65
66  Private Sub Command49_Click()
67  DoCmd.OpenReport "Analysis Rpt", acPreview, , "[AnalysisNo] = " & Me!AnalysisNo
68  End Sub
69
70  Private Sub Command50_Click()
71  ' Print the Bank Balance Data Rpt
72  Dim crt As String
73  crt = "[CustomerNo] = '" & Me!CustomerNo & "'"
74  crt = crt & " AND [RevDate] = #" & Me!RevDate & "#"
75  DoCmd.OpenReport "Bank Balance Data", acViewPreview, , crt
76  End Sub
77
78  Private Sub Command52_Click()
79
80  Const twipsPerInch = 1440
81  DoCmd.OpenReport "Analysis Rpt", acPreview, , "[AnalysisNo] = " & Me!AnalysisNo
82  DoCmd.MoveSize 0, 0, 8 * twipsPerInch, 6 * twipsPerInch
83  End Sub
84
85  Private Sub Command53_Click()
86  If Command53.Caption = "Sequence #" Then
87     Command53.Caption = "Code"
88     Me!AnalysisDtl.Form.OrderBy = "ServiceCode"
89  Else
90     Command53.Caption = "Sequence #"
91     Me!AnalysisDtl.Form.OrderBy = "SeqNo"
92  End If
93
94  End Sub
95
96  Private Sub Command59_Click()
97  Dim crt As String
98  crt = "[CustomerNo] = '" & Me!CustomerNo & "'"
99  crt = crt & " AND [RevDate] = #" & Me!RevDate & "#"
100 D  Cmd.OpenReport "Bank Balance Data By Float", acViewPreview, , crt
101 End Sub
102
```

```
Private Sub C  mmand60_Click()
Dim crt As String
crt = "[Cust  merN  ] = '" & Me!CustomerN   & "'"
crt = crt & " AND [RevDate] = #" & Me!RevDate & "#"
D  Cmd.OpenRep  rt "Bank Balance Data By EC Rate", acViewPreview, , crt

End Sub

Private Sub Command61_Click()
DoCmd.OpenForm "EC Analysis Select"
End Sub

Private Sub CustomerNo_BeforeUpdate(Cancel As Integer)
If Not Me.NewRecord Then
   If MsgBox("You are about to change Customer number." & vbCr & "This will result in
these service codes to be under the new customer." & vbCr & "Is this what you want to
do?", vbOKCancel, "Key Field Change") = vbCancel Then
      DoCmd.CancelEvent
      Me!CustomerNo.Undo
   End If
End If

End Sub

Private Sub CustomerNo_GotFocus()
CustomerNo.Dropdown
End Sub

Private Sub CustomerNo_NotInList(NewData As String, Response As Integer)
Dim Result
Dim msg As String

If NewData = "" Then Exit Sub   ' cleared combo box

msg = "" & NewData & " is not in the Customer file." & vbCr & vbCr & "Do you want to add
' Yes
If MsgBox(msg, vbQuestion + vbYesNo) = vbYes Then
   DoCmd.OpenForm "Customer", , , , acFormAdd, acDialog, NewData
End If

' look for new record added
Result = DLookup("[CustomerNo]", "Customer", "[CustomerNo]= '" & NewData & "'")
If IsNull(Result) Then
   'suppress error message
   Response = acDataErrContinue
   MsgBox "Please enter a new customer number."
Else
   ' they added it
   Response = acDataErrAdded
End If


End Sub

Private Sub Form_Activate()
If CurrentUser() = "Admin" Or CurrentUser = "CarmJr" Then
   DoCmd.ShowToolbar "Menu Bar", acToolbarYes
Else
   DoCmd.ShowToolbar "Menu Bar", acToolbarNo
End If
MaximizeRestoredForm Forms!analysisHdr
End Sub

Private Sub Form_Close()
D  Cmd.OpenForm "Analysis Select  r"
End Sub
```

```
167
168  Private Sub F  rm_Current()
169  If Me.NewRecord Then
170     Cust  merN  .SetF  cus
171  End If
172  End Sub
173
174  Private Sub NegCollectedRate_Exit(Cancel As Integer)
175  If Me.NewRecord Then
176     AnalysisDtl.SetFocus
177  End If
178  End Sub
```

SECTION III – Report Generation

This section describes the black box effect of the invention and discloses the methods and algorithms used to create the desired results. It also lists sample source code of key aspects of the program.

Once the data in the previous two sections has been amassed, it is now ready to be processed. This section illustrates the core functionality of the present invention. A majority of the information revealed by the reports was previously unknown to the clients.

The following lists Key fields which are utilized throughout the Bank Analyzer software:

AvgDlyPos	- the average daily positive balance
FedPct	- the current percentage mandated for the Reserve requirement
AdjBal	- adjusted balance
AvgBal	- average balance
AvgFloat	- average float
ECRate	- earnings credit rate
ReserveReq	- reserve requirement amount
AvailableBalance	- available balance
AvgFloatPerStore	- average float per store
AnalysisNo	- unique indexed number
NegCollectedRate	- negative collected rate, compared with the ECRate

Report: Bank Balance Data (Fig. 2)

Objective: Determine actual Total service charges paid by clients

The TotalServiceCharges field is derived from the CalcServTotal function, which is fed the information from the analyses entered during the dynamic compilation phase. The source code follows, which describes how to derive the actual service charges paid - by bank and with totals (2a), for the client being examined.

```
Public Functi  n CalcServTotal(R As Report)
' Determining actual service charges
```

```
' This function is called for every line displayed and/or printed in the Bank Balance Data report – Fig. 2 in
the disclosure
Dim tmpAmount As Double
Dim msg, crt As String
On Error Resume Next

' select the bank we're examining
crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
crt = crt & " AND [BankCode] = '" & R.BankCode & "'"
crt = crt & " AND [RevDate] = #" & Forms!analysisHdr.RevDate & "#"

' obtain fields from the AnalysisDtl table described in Dynamic Compilation section
tmpAmount = DSum("[Activity]*[UnitPrice]", "AnalysisDtlQ", crt)

' return the answers in variables
ServicesTotal = ServicesTotal + tmpAmount
CalcServTotal = tmpAmount

On Error GoTo 0
End Function
```

Report: Earnings Credit Analysis (Fig. 3)

Objectives:
1. Basis points shaved
2. Projected Fed Fund amount
3. Projected T-Bill amount
4. Projected Income Difference

The report generation process begins with a data entry screen, as shown in Fig. 26. The T-Bill, Fed Funds, Interest Rate and Prime are entered prior to generating the report. These are the default values that the program uses in determining the objectives. These values can be customized by the client to allow for more accurate reporting. Results are revealed in a simple color-coded scheme allowing clients to see "at a glance" exposure to points of concern. (3e)

Exemplary source code follows:

To display the Basis points shaved using color codes: (3e)

```
Public Function ColorCodeBanks
Dim tmpAmt, tmpAmount As Double
PointsShaved.Visible = True
' turn Overdraft balances to Red
Select Case AvgBal
Case Is >= 0
   AvgBal.ForeColor = vbBlack
Case Else
   AvgBal.ForeColor = vbRed
End Select
```

```
' color code how many points are shaved
' constants representing colors are stored in an 'include' file
Select Case PointsShaved
Case Is >= 200
   ' RED --> Worst offenders
   BankCode.ForeColor = vbRed
   BankName.ForeColor = vbRed
   PointsShaved.ForeColor = vbRed
Case 100 To 199
   BankCode.ForeColor = vbCyan
   BankName.ForeColor = vbCyan
   PointsShaved.ForeColor = vbCyan
Case 26 To 99
   BankCode.ForeColor = vbMagenta
   BankName.ForeColor = vbMagenta
   PointsShaved.ForeColor = vbMagenta
Case 1 To 25
   BankCode.ForeColor = vbDarkGreen
   BankName.ForeColor = vbDarkGreen
   PointsShaved.ForeColor = vbDarkGreen
Case Is <= 0
   PointsShaved.Visible = False
   BankCode.ForeColor = vbBlue
   BankName.ForeColor = vbBlue
   PointsShaved.ForeColor = vbBlue
End Select
End Function
```

Objective algorithms:

The Fed Funds Income difference is determined by taking the inputted current Fed Funds percentage and multiplying it by the Available Balance, then subtracting the annualized Earnings Credit amount.

The T-Bill difference is determined by taking the inputted current Treasury Bill rate and multiplying by the Available Balance, then subtracting the annualized Earnings Credit Amount.

The Interest Yield difference is determined by taking the inputted current Interest rate and multiplying by the Available Balance, then subtracting the annualized Earnings Credit Amount.

Key Source Code functions and examples: (3c)

```
FedFundsIncomeDifference=[Forms]![EC Analysis Select]![FedFunds]*[AvailableBal]-([EarningsCredit]*12)
TBillDifference=[Forms]![EC Analysis Select]![Tbill]*[AvailableBal]-([EarningsCredit]*12)
IncomeDifference=[Forms]![EC Analysis Select]![InterestRate]*[AvailableBal]-([EarningsCredit]*12)
```

Source Code display:

```
SELECT IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS ReserveReq,
AnalysisHdr.CustomerNo, AnalysisHdr.BankCode, AnalysisHdr.RevDate, AnalysisHdr.AvgBal,
AnalysisHdr.AvgFloat, AnalysisHdr.FedPct, AnalysisHdr.ECRate, [AvgDlyNeg]+[AdjBal] AS AvgDlyPos,
```

```
AnalysisHdr.AvgDlyNeg, Banks.BankName, Customer.CustomerName, [AvgBal]-[AvgFloat] AS AdjBal,
Banks.City, Banks.State, [AvgDlyPos]-[ReserveReq] AS AvailableBalance, [AvailableBalance]*[ECRate]/1200
AS EarningsCredit, AnalysisHdr.NumberOfStores, [AvgFloat]/[NumberOfStores] AS AvgFloatPerStore,
AnalysisHdr.AnalysisNo, IIf([AvgDlyPos]=0,[AdjBal]-[FedRes],[AvgDlyPos]-[FedRes]) AS AvailableBal,
IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS FedRes,
AnalysisHdr.NegCollectedRate
FROM Customer INNER JOIN (Banks INNER JOIN AnalysisHdr ON Banks.BankCode =

AnalysisHdr.BankCode) ON Customer.CustomerNo = AnalysisHdr.CustomerNo;

Public Function CalcTBillTotal(R As Report)
Dim tmpAmount As Double
If R!TBillIncomeDifference > 0 Then
   tmpAmount = R!TBillIncomeDifference
   TBillDifference = TBillDifference + tmpAmount
End If
If R!FedFundsIncomeDifference > 0 Then
   FedFundsDifference = R!FedFundsIncomeDifference + FedFundsDifference
End If
If R!InterestYieldIncomeDifference > 0 Then
   InterestYieldDifference = R!InterestYieldIncomeDifference + InterestYieldDifference
End If

CalcTBillTotal = tmpAmount
End Function
```

To derive sums and annualized results, the totals are added for each line (corresponding to one client Bank), and then multiplied by 12 for annualization. The report's objectives are supported by displaying the Average Investable income and the Annualized Projected income. (3d)

Report: Earnings Credit And Negative Collected Rates (Fig. 4)

Objective: Match the Earnings Credit on positive balances to the rate imposed on Negative Collected funds.

The report generation process begins with a data entry screen, as shown in Fig. 27.

Objective algorithm:

Display the Negative collected rate on the same report with the Earnings Credit rate.

Exemplary Source code:

Selects and displays the data shown on Fig. 4.

```
SELECT IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS ReserveReq,
AnalysisHdr.CustomerNo, AnalysisHdr.BankCode, AnalysisHdr.RevDate, AnalysisHdr.AvgBal,
AnalysisHdr.AvgFloat, AnalysisHdr.FedPct, AnalysisHdr.ECRate, [AvgDlyNeg]+[AdjBal] AS AvgDlyPos,
AnalysisHdr.AvgDlyNeg, Banks.BankName, Customer.CustomerName, [AvgBal]-[AvgFloat] AS AdjBal,
```

```
Banks.City, Banks.State, [AvgDlyPos]-[ReserveReq] AS AvailableBalance, [AvailableBalance]*[ECRate]/1200
AS EarningsCredit, AnalysisHdr.NumberOfStores, [AvgFloat]/[NumberOfStores] AS AvgFloatPerStore,
AnalysisHdr.AnalysisNo, IIf([AvgDlyPos]=0,[AdjBal]-[FedRes],[AvgDlyPos]-[FedRes]) AS AvailableBal,
IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS FedRes,
AnalysisHdr.NegCollectedRate
FROM Customer INNER JOIN (Banks INNER JOIN AnalysisHdr ON Banks.BankCode =
AnalysisHdr.BankCode) ON Customer.CustomerNo = AnalysisHdr.CustomerNo;
```

Report: Checks Deposits and Float Data (Fig. 5)

Objectives: 1. Determine actual availability date of client's monies (5a)

2. Show abnormal Out Of District activity (5b)

3. Verify immediate availability of On Us activity (5c)

Exemplary Source Code for report output:

The INNER JOIN ensures proper bank and client information extracted.

```
SELECT IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS ReserveReq,
AnalysisHdr.CustomerNo, AnalysisHdr.BankCode, AnalysisHdr.RevDate, AnalysisHdr.AvgBal,
AnalysisHdr.AvgFloat, AnalysisHdr.FedPct, AnalysisHdr.ECRate, [AvgDlyNeg]+[AdjBal] AS AvgDlyPos,
AnalysisHdr.AvgDlyNeg, Banks.BankName, Customer.CustomerName, [AvgBal]-[AvgFloat] AS AdjBal,
Banks.City, Banks.State, [AvgDlyPos]-[ReserveReq] AS AvailableBalance, [AvailableBalance]*[ECRate]/1200
AS EarningsCredit, AnalysisHdr.NumberOfStores, [AvgFloat]/[NumberOfStores] AS AvgFloatPerStore,
AnalysisHdr.AnalysisNo, IIf([AvgDlyPos]=0,[AdjBal]-[FedRes],[AvgDlyPos]-[FedRes]) AS AvailableBal,
IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS FedRes,
AnalysisHdr.NegCollectedRate, Customer.PDL1, Customer.PDL2, Customer.PDL3, Customer.PDL4,
AnalysisHdr.ChecksOutDist, AnalysisHdr.ChecksInDist, AnalysisHdr.MixedDeposits,
AnalysisHdr.DepositTickets, AnalysisHdr.ChecksOnUs,
IIf([TotalChecksFloated]>0,[AvgFloat]/[TotalChecksFloated]*30,0) AS AvgCheckSize,
[ChecksOutDist]+[ChecksInDist]+[MixedDeposits] AS TotalChecksFloated, AnalysisHdr.DepCost,
AnalysisHdr.BAICost, AnalysisHdr.DepReconCost, AnalysisHdr.NightBagsCost
FROM Customer INNER JOIN (Banks INNER JOIN AnalysisHdr ON Banks.BankCode =
AnalysisHdr.BankCode) ON Customer.CustomerNo = AnalysisHdr.CustomerNo
ORDER BY AnalysisHdr.AvgFloat DESC;
```

Sample VBA Code: Checks Deposits and Float Data

```
Code
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Report_Checks Deposits and Float Data"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub GroupF    ter1_Print(Cancel As Integer, PrintCount As Integer)
'
'
End Sub

```

```
Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
' this is d  ne in the Activate Event n  w
'Dim tDT, tMD, tCOU, tCID, tCOD As D  uble
'Call CalcCDFData(Rep  rt, tDT, tMD, tCOU, tCID, tCOD)
'DepositTickets = tDT
'MixedDep  sits = tMD
'ChecksOnUs = tCOU
'ChecksInDist = tCID
'ChecksOutDist = tCOD
End Sub

Private Sub Report_Activate()
' generate the activity data
Call CalcCDFData
End Sub

Private Sub Report_Deactivate()
Application.Echo True
End Sub

Private Sub ReportFooter_Print(Cancel As Integer, PrintCount As Integer)
Dim tDT, tMD, tCOU, tCID, tCOD As Double
Call GetCheckDepTotals(Report, tDT, tMD, tCOU, tCID, tCOD)
SumDepositTickets = tDT
SumMixedDeposits = tMD
SumChecksOnUs = tCOU
SumChecksInDist = tCID
SumChecksOutDist = tCOD
End Sub
```

Report: Breakdown By Service Charge Groups (Fig. 6)

Objectives: 1. Categorize Depository Costs (6a)

2. Categorize Check Costs (6b)

3. Categorize Account Maintenance Costs (6c)

Sample VBA Code: Breakdown By Service Charge Groups

For the report output:

```
SELECT IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS ReserveReq,
AnalysisHdr.CustomerNo, AnalysisHdr.BankCode, AnalysisHdr.RevDate, AnalysisHdr.AvgBal, AnalysisHdr.AvgFloat,
AnalysisHdr.FedPct, AnalysisHdr.ECRate, [AvgDlyNeg]+[AdjBal] AS AvgDlyPos, AnalysisHdr.AvgDlyNeg, Banks.BankName,
Customer.CustomerName, [AvgBal]-[AvgFloat] AS AdjBal, Banks.City, Banks.State, [AvgDlyPos]-[ReserveReq] AS
AvailableBalance, [AvailableBalance]*[ECRate]/1200 AS EarningsCredit, AnalysisHdr.NumberOfStores,
[AvgFloat]/[NumberOfStores] AS AvgFloatPerStore, AnalysisHdr.AnalysisNo, IIf([AvgDlyPos]=0,[AdjBal]-
[FedRes],[AvgDlyPos]-[FedRes]) AS AvailableBal,
IIf([AvgDlyPos]=0,([FedPct]/100)*[AdjBal],([FedPct]/100)*[AvgDlyPos]) AS FedRes, AnalysisHdr.NegCollectedRate,
Customer.PDL1, Customer.PDL2, Customer.PDL3, Customer.PDL4, AnalysisHdr.ChecksOutDist, AnalysisHdr.ChecksInDist,
AnalysisHdr.MixedDeposits, AnalysisHdr.DepositTickets, AnalysisHdr.ChecksOnUs,
IIf([TotalChecksFloated]>0,[AvgFloat]/[TotalChecksFloated]*30,0) AS AvgCheckSize,
[ChecksOutDist]+[ChecksInDist]+[MixedDeposits] AS TotalChecksFloated, AnalysisHdr.DepCost, AnalysisHdr.BAICost,
AnalysisHdr.DepReconCost, AnalysisHdr.NightBagsCost
FROM Customer INNER JOIN (Banks INNER JOIN AnalysisHdr ON Banks.BankCode = AnalysisHdr.BankCode) ON
Customer.CustomerNo = AnalysisHdr.CustomerNo
ORDER BY AnalysisHdr.AvgFloat DESC;
```

For the calculations:

Code

```
VERSION 1.0 CLASS
```

```
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Rep  rt_Breakdown By Service Charge Groups"
Attribute VB_Gl  balNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
Dim crt As String
crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
crt = crt & " AND [BankCode] = '" & Report.BankCode & "'"
crt = crt & " AND [RevDate] = #" & Forms!analysisHdr.RevDate & "#"

AnnualServiceCharges = DSum("[ExtCharge]", "AnalysisDtlQ", crt) * 12
' Calculate the totals
DepositCost = CalcCostTotal(crt, 2) * 12
CheckDepCost = CalcCostTotal(crt, 1) * 12
BalInfoCost = CalcCostTotal(crt, 3) * 12
TotalCost = DepositCost + CheckDepCost + BalInfoCost
' derive %'s
DepositCostPer = DepositCost / AnnualServiceCharges
CheckDepCostPer = CheckDepCost / AnnualServiceCharges
BalInfoCostPer = BalInfoCost / AnnualServiceCharges
TotalCostPEr = TotalCost / AnnualServiceCharges

End Sub

Private Sub ReportFooter_Print(Cancel As Integer, PrintCount As Integer)
Dim crt As String
crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
crt = crt & " AND [RevDate] = #" & Forms!analysisHdr.RevDate & "#"

SAnnualServiceCharges = DSum("[ExtCharge]", "AnalysisDtlQ", crt) * 12
' Calculate the totals
SDepositCost = CalcCostTotal(crt, 2) * 12
SCheckDepCost = CalcCostTotal(crt, 1) * 12
SBalInfoCost = CalcCostTotal(crt, 3) * 12
STotalCost = SDepositCost + SCheckDepCost + SBalInfoCost
' derive %'s
SDepositCostPer = SDepositCost / SAnnualServiceCharges
sCheckDepCostPer = SCheckDepCost / SAnnualServiceCharges
SBalInfoCostPer = SBalInfoCost / SAnnualServiceCharges
STotalCostPer = STotalCost / SAnnualServiceCharges

End Sub
```

Report: Breakdown By User Supplied Groups (Fig. 7)

Objective: Categorize any user defined bank cost

To achieve this special objective, an additional Category table, shown in Fig. 28, is added to the database. This table works in conjunction with the Service Codes table shown in Fig. 17. A Category Select Form is also used in this process. A screen display of the form is shown in Fig. 29. The example in Fig. 29 shows categories, 60 (night bags), 62 (rolled coins) and 64 (return checks) as being selected.

Sample VBA Code: Breakdown By Service Charges Variable

C_de

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Report_Breakdown By Service Charges Variable"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
Dim crt, CatCode1, CatCode2, CatCode3 As String
crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
crt = crt & " AND [BankCode] = '" & Report.BankCode & "'"
crt = crt & " AND [RevDate] = #" & Forms!analysisHdr.RevDate & "#"

CatCode1 = Forms!CatSelect!CatCode1
CatCode2 = Forms!CatSelect!CatCode2
CatCode3 = Forms!CatSelect!CatCode3
AnnualServiceCharges = DSum("[ExtCharge]", "AnalysisDtlQ", crt) * 12
' Calculate the totals
' Var names are carried over from base report, they're actually Cat1, Cat2, Cat3
DepositCost = CalcCostTotal(crt, CatCode1) * 12
CheckDepCost = CalcCostTotal(crt, CatCode2) * 12
BalInfoCost = CalcCostTotal(crt, CatCode3) * 12
TotalCost = DepositCost + CheckDepCost + BalInfoCost
' derive %'s
DepositCostPer = DepositCost / AnnualServiceCharges
CheckDepCostPer = CheckDepCost / AnnualServiceCharges
BalInfoCostPer = BalInfoCost / AnnualServiceCharges
TotalCostPEr = TotalCost / AnnualServiceCharges

End Sub

Private Sub PageHeader_Print(Cancel As Integer, PrintCount As Integer)
Dim crt As String

crt = "[CategoryCode] = " & Forms!CatSelect!CatCode1
CatLabel1 = DLookup("[Description]", "Category", crt)

crt = "[CategoryCode] = " & Forms!CatSelect!CatCode2
CatLabel2 = DLookup("[Description]", "Category", crt)

crt = "[CategoryCode] = " & Forms!CatSelect!CatCode3
CatLabel3 = DLookup("[Description]", "Category", crt)

End Sub


Private Sub Report_Activate()
DoCmd.Maximize
End Sub

Private Sub ReportFooter_Print(Cancel As Integer, PrintCount As Integer)
Dim crt, CatCode1, CatCode2, CatCode3 As String
crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
crt = crt & " AND [RevDate] = #" & F  rms!analysisHdr.RevDate & "#"

CatCode1 = F  rms!CatSelect!CatCode1
CatCode2 = F  rms!CatSelect!CatCode2
CatC  de3 = Forms!CatSelect!CatCode3
SAnnualServiceCharges = DSum("[ExtCharge]", "AnalysisDtlQ", crt) * 12
```

```
64  ' Calculate the totals
65  SDep  sitCost = CalcCostT  tal(crt, CatCode2) * 12
66  SCheckDepCost = CalcCostTotal(crt, CatCode1) * 12
67  SBalInf  C  st = CalcCostT  tal(crt, CatCode3) * 12
68  ST  talC  st = SDep  sitC  st + SCheckDepC  st + SBalInfoC  st
69  ' derive %'s
70  SDepositCostPer = SDepositCost / SAnnualServiceCharges
71  sCheckDepCostPer = SCheckDepCost / SAnnualServiceCharges
72  SBalInfoCostPer = SBalInfoCost / SAnnualServiceCharges
73  STotalCostPer = STotalCost / SAnnualServiceCharges
74  End Sub
```

This report uses the same query as Breakdown by Service Charge groups for Report selection. However, in this report, the users can supply their own categories.

Report: Unit Price Summary (Fig. 8)

Objective: Show every service charge assessed to a client

Sample code for the report output:

```
SELECT AnalysisHdr.CustomerNo, AnalysisHdr.BankCode, AnalysisHdr.RevDate, AnalysisDtl.ServiceCode,
Banks.BankName, Banks.City, Banks.State, Customer.CustomerName, AnalysisDtl.Activity,
AnalysisDtl.UnitPrice, ServiceCodes.ServiceDescription, [UnitPrice]*[Activity]/[Per] AS ExtAmt,
ServiceCodes.Per, AnalysisDtl.ConCharge, AnalysisDtl.ConDate
FROM ServiceCodes INNER JOIN (((AnalysisDtl INNER JOIN Banks ON AnalysisDtl.BankCode =
Banks.BankCode) INNER JOIN Customer ON AnalysisDtl.CustomerNo = Customer.CustomerNo) INNER JOIN
AnalysisHdr ON (AnalysisHdr.RevDate = AnalysisDtl.RevDate) AND (AnalysisHdr.BankCode =
AnalysisDtl.BankCode) AND (AnalysisHdr.CustomerNo = AnalysisDtl.CustomerNo) AND
(Customer.CustomerNo = AnalysisHdr.CustomerNo) AND (Banks.BankCode = AnalysisHdr.BankCode)) ON
ServiceCodes.ServiceCode = AnalysisDtl.ServiceCode
WHERE (((AnalysisDtl.ServiceCode)<"95000"));
```

This report features a customizable form to further allow varying degrees of results. A sample screen display of the form is shown in Fig. 30. The following source code controls the flexible data entry form shown in Fig. 30. It allows for the user to selectively choose any combination of clients, banks, and/or service charges. It also allows for entry of date ranges and dollar figures, and provides a method of highlighting particular information once the report is generated.

Code

```
1  VERSION 1.0 CLASS
2  BEGIN
3    MultiUse = -1  'True
4  END
5  Attribute VB_Name = "Form_AnalysisRpt Select"
6  Attribute VB_GlobalNameSpace = False
7  Attribute VB_Creatable = True
8  Attribute VB_PredeclaredId = True
9  Attribute VB_Exp  sed = False
10 Opti  n Compare Database
11 Opti  n Explicit
12
13 Private Sub BankCheck_Click()
```

```
14  ' if the user wants every bank, don't let them select from the list box
15  Dim ctl As Control, itm As Variant
16  Set ctl = Me!BankList
17  If BankCheck Then
18     ' turn   ff anything they had selected
19     For Each itm In ctl.ItemsSelected
20        ctl.Selected(itm) = False
21     Next itm
22     ' turn the Bank list box and counter OFF
23     BankList.Enabled = False
24     BanksSelected.Visible = False
25     ' turn the OK button ON
26     Command16.Enabled = True
27  Else
28     ' turn the Bank List box and counter ON
29     BankList.Enabled = True
30     BanksSelected.Visible = True
31  End If
32
33  End Sub
34
35  Private Sub BankList_Click()
36
37  ' Controls the OK button, and makes sure we don't build too big of a query
38  ' This procedure is also triggered when the Customer & service boxes are clicked
39  Dim totSelected As Integer
40  totSelected = BankList.ItemsSelected.Count + CustomerList.ItemsSelected.Count +
    ServiceList.ItemsSelected.Count
41
42  Select Case totSelected
43     Case 0
44        If BankCheck Or ServiceCheck Or CustomerCheck Then
45           Command16.Enabled = True
46        Else
47           Command16.Enabled = False
48        End If
49
50     Case Is <= 30   ' Access limit is 40, I'm not taking any chances with memory
51        Command16.Enabled = True
52     Case Else
53        Command16.Enabled = False
54        MsgBox "The maximum number of custom selections is 30." & vbCr & "Please
    de-select some items and try again.", vbOKOnly, "Too many items selected"
55  End Select
56
57  End Sub
58
59  Private Sub Command16_Click()
60  'October 99 Carmen DeLeo, Jr.
61  ' crtAll will be the concatenation of all the built criteria
62  Dim Q As QueryDef, db As Database
63  Dim highlightCustNo, crtBank, crtService, crtCustomer, crtAll As String
64  Dim ctlBank, ctlService, ctlCustomer As Control
65  Dim itm As Variant
66
67  ' get the bank(s) selected
68  ' ************************
69  Set ctlBank = Me![BankList]
70  If Not BankCheck Then   ' If they didn't select All banks, what did they select?
71     For Each itm In ctlBank.ItemsSelected
72        'Build the criteria string
73        If Len(crtBank) = 0 Then
74           crtBank = "[BankCode] = '" & ctlBank.ItemData(itm) & "'"
75        Else
76           crtBank = crtBank & " OR [BankCode] = '" & ctlBank.ItemData(itm) & "'"
77        End If
```

```
    Next itm
    If Len(crtBank) = 0 Then
      MsgB  x "No Banks selected."
      Exit Sub
    End If
  Else
    ' give us every bank
    crtBank = "[BankCode] > '0'"
  End If

  ' get the Services
  ' ****************
  Set ctlService = Me![ServiceList]
  If Not ServiceCheck Then ' If they didn't select All services, what did they select?
    For Each itm In ctlService.ItemsSelected
      ' Build it
      If Len(crtService) = 0 Then
        crtService = "[ServiceCode] = '" & ctlService.ItemData(itm) & "'"
      Else
        crtService = crtService & " OR [ServiceCode] = '" & ctlService.ItemData(itm) & "'"
      End If
    Next itm
    If Len(crtService) = 0 Then
      MsgBox "No service codes selected."
      Exit Sub
    End If
  Else
    ' Give us every service code
    crtService = "[ServiceCode] > '0'"
  End If

  ' get the customer accounts
  ' *************************
  Set ctlCustomer = Me![CustomerList]
  If Not CustomerCheck Then
    For Each itm In ctlCustomer.ItemsSelected
      If Len(crtCustomer) = 0 Then
        crtCustomer = "[CustomerNo] = '" & ctlCustomer.ItemData(itm) & "'"
      Else
        crtCustomer = crtCustomer & " OR [CustomerNo] = '" & ctlCustomer.ItemData(itm)
    & "'"
      End If
    Next itm
    If Len(crtCustomer) = 0 Then
      MsgBox "No customers selected."
      Exit Sub
    End If
  Else
    ' give us every customer
    crtCustomer = "[CustomerNo] > '0'"
  End If

  ' Put it all together for banks, services, clients
  crtAll = "(" & crtBank & ") AND (" & crtService & ") AND (" & crtCustomer & ")"
  ' tack on date and dollar ranges
  crtAll = crtAll & " AND [RevDate] >= #" & Me!StartDate & "# AND [RevDate] <= #" &
    Me!EndDate & "#"
  crtAll = crtAll & " AND [UnitPrice] >= " & Val(Me!StartDollar) & " AND [UnitPrice] <= " &
    Val(Me!EndDollar)

  ' Run the rep  rt with built criteria
  If Sh  wSummaryInfo Then
    D  Cmd.OpenRep  rt "Analysis By Unit Price Summary", acViewPreview, , crtAll
  Else
    DoCmd.OpenRep  rt "Analysis By Unit Price", acViewPreview, , crtAll
  End If
```

```
End Sub

Private Sub Command17_Click()
    D  Cmd.Cl  se acF  rm, "AnalysisRpt Select"
End Sub

Private Sub CustomerCheck_Click()
' if the user wants every Customer, don't let them select from the list box
Dim ctl As Control, itm As Variant
Set ctl = Me!CustomerList

If CustomerCheck Then
   ' turn off anything they had selected
   For Each itm In ctl.ItemsSelected
      ctl.Selected(itm) = False
   Next itm
   ' turn the Customer List box and counter OFF
   CustomerList.Enabled = False
   CustomersSelected.Visible = False
   ' turn the OK Button ON
   Command16.Enabled = True
Else
   ' turn the Customer List box and counter ON
   CustomerList.Enabled = True
   CustomersSelected.Visible = True
End If
End Sub

Private Sub CustomerList_Click()
BankList_Click
End Sub

Private Sub ServiceCheck_Click()
Dim ctl As Control, itm As Variant

' if the user wants every Service, don't let them select from the list box
Set ctl = Me!ServiceList
If ServiceCheck Then
   ' turn off anything they had selected
   For Each itm In ctl.ItemsSelected
      ctl.Selected(itm) = False
   Next itm
   ' turn off the whole List box and counter
   ServiceList.Enabled = False
   ServicesSelected.Visible = False

   ' put the OK button back ON
   Command16.Enabled = True
Else
   ' turn the list box and counter back ON
   ServiceList.Enabled = True
   ServicesSelected.Visible = True
End If
End Sub

Private Sub ServiceList_Click()
BankList_Click
End Sub
```

Once the criteria are set, the report is generated.

The following sample code generates the actual report of Unit Price Summary (Fig. 8) and also controls the highlighting of particular information as specified by the user above.

Code

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Rep  rt_Analysis By Unit Price"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
If Not IsNull(Forms![AnalysisRpt Select]!highlightCustNo) Then
   If CustomerNo = Nz(Forms![AnalysisRpt Select]!highlightCustNo) Then
      CustomerNo.ForeColor = 255
      CustomerName.ForeColor = 255
      BankName.ForeColor = 255
      BankCode.ForeColor = 255
      City.ForeColor = 255
      State.ForeColor = 255
      Activity.ForeColor = 255
      UnitPrice.ForeColor = 255
      ExtAmt.ForeColor = 255
      RevDate.ForeColor = 255
   Else
      CustomerNo.ForeColor = 0
      CustomerName.ForeColor = 0
      BankName.ForeColor = 0
      BankCode.ForeColor = 0
      City.ForeColor = 0
      State.ForeColor = 0
      Activity.ForeColor = 0
      UnitPrice.ForeColor = 0
      ExtAmt.ForeColor = 0
      RevDate.ForeColor = 0
   End If
End If
End Sub

Private Sub Report_Activate()
DoCmd.Maximize
If Forms![AnalysisRpt Select]!ShowNames Then
   CustomerName.Visible = True
Else
   CustomerName.Visible = False
End If

End Sub

Private Sub ReportHeader_Print(Cancel As Integer, PrintCount As Integer)
Dim hCust As String
hCust = Nz(Forms![AnalysisRpt Select]!highlightCustNo, "")
If Len(hCust) > 0 Then
   HighlightedCustomer = Forms![AnalysisRpt Select]!highlightCustNo
   HighlightedCustomerName = DLookup("[CustomerName]", "Customer", "[CustomerNo]
= '" & hCust & "'")
Else
   Label36.Visible = False
End If
End Sub
```

Report: Deposit Ticket Costs (Fig. 9)

C de

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Report_Deposit Ticket Costs"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub GroupFooter1_Print(Cancel As Integer, PrintCount As Integer)
'
'
End Sub

Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
' setup our temp variables
Dim tD, tDepC, tBAI, tBAIO, tBAIC, tDR, tDRC, tDRO, tDepR, tDepRC, tNB, tNBC As Double
' go get totals
Call CalcDepTicketData(Report, tBAI, tBAIO, tDR, tDRO, tDepR, tDepRC, tNB)
' assign fields on report
'Deposit = tD
'DepCost = tDepC    *** Commented fields are calculated in Activate proc
BAI = tBAI
'BAICost = tBAIC
BAIOther = tBAIO
DepRecon = tDR
'DepReconCost = tDRC
DepReconOther = tDRO
DepReorder = tDepR
DepReorderCost = tDepRC
NightBags = tNB
'NightBagsCost = tNBC
On Error Resume Next
If Loss1 > 0 Then Loss1.ForeColor = vbRed Else Loss1.ForeColor = vbBlack
If Loss2 > 0 Then Loss2.ForeColor = vbRed Else Loss2.ForeColor = vbBlack
If Loss3 > 0 Then Loss3.ForeColor = vbRed Else Loss3.ForeColor = vbBlack
If Loss4 > 0 Then Loss4.ForeColor = vbRed Else Loss4.ForeColor = vbBlack
On Error GoTo 0
End Sub

Private Sub Report_Activate()
Call WriteDepTicketData
End Sub

Private Sub Report_Deactivate()
Application.Echo True
End Sub

Private Sub ReportFooter_Print(Cancel As Integer, PrintCount As Integer)
Dim t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12 As Double
Call GetDepTicketTotals(Report, t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12)
SumDeposit = t1
SumDepCost = t2
SumBAI = t3
SumBAIC  st = t4
SumBAIOther = t5
SumDepRecon = t6
SumDepReconCost = t7
```

```
62  SumDepReconOther = t8
63  SumDepReorder = t9
64  SumDepReorderCost = t10
65  SumNightBags = t11
66  SumNightBagsCost = t12
67
68
69  End Sub
70
71  Private Sub ReportHeader_Print(Cancel As Integer, PrintCount As Integer)
72  InitVars
73  End Sub
```

Multiplying the SumDeposit field in line 55 above by 12 provides the projected Annual cost incurred for one year's worth of deposit costs. (9a)

The BreakEvenDeposit (9a) is determined by the following formula:

((([DepCost]+[BAICost]+[DepReconCost]+[NightBagsCost])/[DepositTickets])*365)/[Forms]![DepCost Select]![ReturnRate]

where the ReturnRate is a value inputted by the user (9e) which represents a typical net profit percentage for the particular client.

The values referred to in (9c) are stored in the client's profile in the database.

Lines 37 through 40 of the code determine the appropriate color for values referenced in (9d).

Report: Cash Activity Costs (Fig. 10)

Sample VBA Code for CashActivity Costs

```
Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
Dim crt As String
crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
crt = crt & " AND [BankCode] = '" & Report.BankCode & "'"
crt = crt & " AND [RevDate] = #" & Forms!analysisHdr.RevDate & "#"

DepPerUnitCost = 0  ' reset it- so it doesn't dupe
' Cash Deposit Per 1000
CashDepPer1000 = CalcDepTotal(crt, 101)
CashCostPer1000 = CalcCostTotal(crt, 101)
If CashDepPer1000 <> 0 Then
   ' avg unit cost has to multiply back by Per1000
   DepPerUnitCost = (CashCostPer1000 / CashDepPer1000) * 1000      Fig. 10, item 10b
End If
' Cash Strapped
DepStrappedUnitCost = 0
CashDepStrapped = CalcDepTotal(crt, 102)
DepStrappedCost = CalcCostTotal(crt, 102)
If DepStrappedCost <> 0 Then
   DepStrappedUnitCost = DepStrappedCost / CashDepStrapped
End If
```

```
' Cash Loose
DepLooseUnitCost = 0
CashDepLoose = CalcDepTotal(crt, 103)
DepLooseCost = CalcCostTotal(crt, 103)
If CashDepLoose <> 0 Then
   DepLooseUnitCost = DepLooseCost / CashDepLoose
End If
' Other Cash costs
OtherCashUnitCost = 0   ' so it doesn't dupe
OtherCash = CalcDepTotal(crt, 104)
OtherCashTotal = CalcCostTotal(crt, 104)
If OtherCash <> 0 Then
   OtherCashUnitCost = OtherCashTotal / OtherCash
End If
' Last few columns
'NightBagsCost = CalcCostTotal(crt, 60)
AvgCashPerStore = CashDepPer1000 / NumberOfStores
CashOrderCost = CalcCostTotal(crt, 61)
RolledCoinCost = CalcCostTotal(crt, 62)                (Fig. 10, item 10b)
BoxCoinCost = CalcCostTotal(crt, 63)

End Sub
```

Multiplying the results from (10b) * 12 yields the results in (10c).

Report: Banking Activity (Fig. 11)

Objective: Summarize information

This report summarizes information presented in other reports. It provides one place to view Deposit Ticket costs, BAI Detail costs, Dep Recon costs, Night Bag costs, and Rolled Coin Costs.

Code

```
1  VERSION 1.0 CLASS
2  BEGIN
3    MultiUse = -1  'True
4  END
5  Attribute VB_Name = "Report_Banking Activity"
6  Attribute VB_GlobalNameSpace = False
7  Attribute VB_Creatable = True
8  Attribute VB_PredeclaredId = True
9  Attribute VB_Exposed = False
10 Option Compare Database
11 Option Explicit
12
13 Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
14 Dim crt As String
15 crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
16 crt = crt & " AND [BankCode] = '" & Report.BankCode & "'"
17 crt = crt & " AND [RevDate] = #" & F  rms!analysisHdr.RevDate & "#"
18
19 Dep  sitTickets = CalcDepT  tal(crt, 20)
20 BAIDetail = CalcDepT  tal(crt, 40)
```

```
21  D  pRecon = CalcDepTotal(crt, 45)
22  DepErrors = CalcDepT  tal(crt, 46)
23  ChecksDep  sited = CalcDepT  tal(crt, 1)
24  CashDep  sited = CalcDepTotal(crt, 101)
25  NightBags = CalcDepT  tal(crt, 60)
26  CashOrdered = CalcDepTotal(crt, 61)
27  RolledCoin = CalcDepTotal(crt, 62)
28  BoxCoin = CalcDepTotal(crt, 63)
29  ReturnChecks = CalcDepTotal(crt, 64)
30  ReturnChecksReDep = CalcDepTotal(crt, 65)
31  ReturnCheckBuyBack = CalcDepTotal(crt, 66)
32  NSFUNCOD = CalcDepTotal(crt, 67)
33
34
35  End Sub
36
37  Private Sub Report_Activate()
38  DoCmd.Maximize
39  End Sub
40
41  Private Sub ReportFooter_Print(Cancel As Integer, PrintCount As Integer)
42  ' Carmen J. DeLeo July 8, 1999
43  Dim crt As String
44  ' No bank criteria on Sums
45  crt = "[CustomerNo] = '" & Forms!analysisHdr.CustomerNo & "'"
46  crt = crt & " AND [RevDate] = #" & Forms!analysisHdr.RevDate & "#"
47  SDepositTickets = CalcDepTotal(crt, 20)
48  SBAIDetail = CalcDepTotal(crt, 40)
49  SDepRecon = CalcDepTotal(crt, 45)
50  SDepErrors = CalcDepTotal(crt, 46)
51  SChecksDeposited = CalcDepTotal(crt, 1)
52  SCashDeposited = CalcDepTotal(crt, 101)
53  SNightBags = CalcDepTotal(crt, 60)
54  SCashOrdered = CalcDepTotal(crt, 61)
55  SRolledCoin = CalcDepTotal(crt, 62)
56  SBoxCoin = CalcDepTotal(crt, 63)
57  SReturnChecks = CalcDepTotal(crt, 64)
58  SReturnChecksRedep = CalcDepTotal(crt, 65)
59  SReturnCheckBuyback = CalcDepTotal(crt, 66)
60  SNSFUNCOD = CalcDepTotal(crt, 67)
61
62  End Sub
```

CalcDepTotal function is used in many of the reports:

```
Public Function CalcDepTotal(crt As String, CatCode)
Dim tmpAmount, tmpCost As Double
Dim msg, crtAll As String

crtAll = crt & " AND [CategoryCode] = " & CatCode
tmpAmount = Nz(DSum("[Activity]", "AnalysisDtlSubTotalsQ", crtAll), 0)
CalcDepTotal = tmpAmount

End Function
```

This function is vital to determining the totals stored in the underlying Analysis Detail tables. As illustrated by the source code, this routine processes a category, then returns the total based on the Service and Category tables.

Report: Analysis By Unit Price (Fig. 12)

The source code is equivalent to the above report, except for the output.

Note: The Detail section of this report is not hidden, thereby showing the supporting data.

```
SELECT AnalysisHdr.CustomerNo, AnalysisHdr.BankCode, AnalysisHdr.RevDate, AnalysisDtl.ServiceCode,
Banks.BankName, Banks.City, Banks.State, Customer.CustomerName, AnalysisDtl.Activity,
AnalysisDtl.UnitPrice, ServiceCodes.ServiceDescription, [UnitPrice]*[Activity]/[Per] AS ExtAmt,
ServiceCodes.Per, AnalysisDtl.ConCharge, AnalysisDtl.ConDate
FROM ServiceCodes INNER JOIN (((AnalysisDtl INNER JOIN Banks ON AnalysisDtl.BankCode =
Banks.BankCode) INNER JOIN Customer ON AnalysisDtl.CustomerNo = Customer.CustomerNo) INNER JOIN
AnalysisHdr ON (AnalysisHdr.RevDate = AnalysisDtl.RevDate) AND (AnalysisHdr.BankCode =
AnalysisDtl.BankCode) AND (AnalysisHdr.CustomerNo = AnalysisDtl.CustomerNo) AND
(Customer.CustomerNo = AnalysisHdr.CustomerNo) AND (Banks.BankCode = AnalysisHdr.BankCode)) ON
ServiceCodes.ServiceCode = AnalysisDtl.ServiceCode
WHERE (((AnalysisDtl.ServiceCode)<"95000"));
```

DETERMINING PARTICULAR CATEGORY COSTS

Fig. 31 and Fig. 32 show excerpts from charts that disclose the method for determining a particular category's cost, using Night Bags as an example. This functionality is used frequently throughout the software.

The following code will determine the total cost. It assumes that the statement data has been keyed in, during the Dynamic Compilation phase.

```
Public Function CalcCostTotal(crt, CatCode)
' return the cost of a group of service charges
Dim tmpAmount, tmpCost As Double
Dim msg, crtAll As String

crtAll = crt & " AND [CategoryCode] = " & CatCode
tmpAmount = Nz(DSum("[ExtCharge]", "AnalysisDtlSubTotalsQ", crtAll), 0)
CalcCostTotal = tmpAmount
End Function
```

This algorithm applies directly to Breakdown By Service Charge, and by User Supplied Groups.

SECTION IV – ILLUMINATION (Fig. 13 and Fig. 14)

Fig. 33 is a screen display of a Working Sheet Form that is used to summarize ServiceDescription items, and their actual costs and potential cost savings.

The Bank Analyzer described above allows a client to obtain information that they have never been able to access. In doing so, it permits the client to eliminate wasteful banking transactions, reduce the amount of transactions and obtain a reduction in bank service charges and other charges and fees.

The Bank Analyzer is preferably used by a third party entity, who has a plurality of clients who provide the third party entity with their respective bank information for analysis. The third party entity may then build up a historical database of bank charges and credits from a plurality of different banks, and can then provide accurate comparisons of a specific client's bank analysis. However, the Bank Analyzer may also be sold or licensed by the third party entity to a client, along with a database of current bank data so that the client may perform their own analyses.

The present invention may be implemented with any combination of hardware and software. The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer useable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the mechanisms of the present invention. The article of manufacture can be included as part of a computer system or sold separately.

It will be appreciated by those skilled in the art that changes could be made to the embodiments described above without departing from the broad inventive concept thereof. It is understood, therefore, that this invention is not limited to the particular embodiments disclosed, but it is intended to cover modifications within the spirit and scope of the present invention.

What is claimed is: